POLYTECH LILLE

Université de Lille 1 SCIENCES ET TECHNOLOGIES

Rapport de Projet de Fin d'Etudes

# Juke-Box Multi-Pièces

Année Universitaire 2015/2016
Projet de Fin d'Etudes IMA5


**Elèves**
Julien HERIN
Alexandre JOUY

**Encadrants école**
Thomas VANTROYS
Alexandre BOE
Rodolphe ASTORI

**Rédigé par Julien HERIN**

# Table des matières

# Introduction

Ce rapport a pour but d'exposer le travail réalisé dans le cadre du Projet de Fin d'Etudes IMA5.

Nous avons choisi de réaliser un Juke-Box Multi-pièces moderne. Ce choix a été motivé tout d'abord par le fait que nous souhaitions réaliser un réseau de haut-parleurs au sein d'une maison permettant de streamer différentes musiques dans différentes pièces. Nous avons aussi beaucoup d'affinités avec la programmation, l'électronique, les innovations technologiques et le matériel audio en général. Nous voyions donc dans ce sujet l'occasion d'exprimer assez librement notre créativité.

Nous commencerons tout d'abord par présenter le contexte du projet et par décrire le cahier des charges que nous nous sommes fixé. Nous parlerons ensuite de la méthode de travail utilisée. Ensuite, nous exposerons la partie software – qui sera bien plus explicitée dans le rapport d'Alexandre JOUY - puis hardware en évoquant à chaque fois le résultat obtenu, les problèmes rencontrés, et les éventuelles possibilités d'amélioration.

# 1 Cahier des charges

## 1.1 Contexte

Depuis déjà deux ou trois ans, les enceintes multi-room se font de plus en plus connaître sur le marché de l'audio. Bose, Samsung, Sonos ou encore fin 2015 Google proposent des haut-parleurs connectés capables de jouer de la musique streamée depuis un ordinateur, une tablette ou un Smartphone.

Cependant, il faudra débourser pas moins d'une centaine d'euros par haut-parleur, pour les dispositifs d'entrée de gamme. De plus, il faudra se contenter de la qualité audio qui nous sera fournie avec le dispositif puisque la modularité y est inexistante.

## 1.2 Objectifs du projet

Le but de ce projet sera donc de proposer les mêmes fonctionnalités à l'utilisateur tout en profitant pleinement d'une modularité sans limites (choix de la qualité audio, du circuit d'alimentation, du design, etc.). De plus, nous proposerons un tutoriel type instructables.com pour fournir les instructions nécessaires au bon fonctionnement de base (streaming, application PC/Smartphone, etc.).

## 1.3 Définition du besoin

Ce projet doit :

- Être modulaire
- Avoir un coût maîtrisé et attractif
- Être Open Source
- Être suffisamment documenté pour être « Do It Yourself »

Pour les haut-parleurs multi-room du marché, si on regarde un graphe de type « radar » avec les caractéristiques suivantes :

- Coût
- Flux audio différents
- Modularité
- Open Source
- Qualité Audio

On a quasiment ceci :

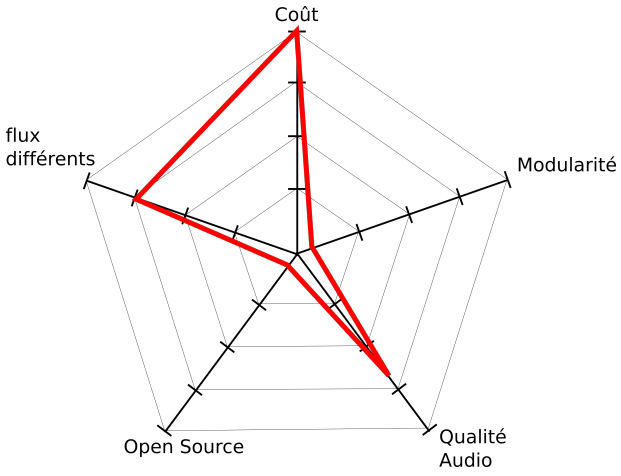



Figure 1 : Graphe radar pour Haut-parleur multi-room Samsung, Bose, Sonos, etc. (incluant amplificateur + HP)

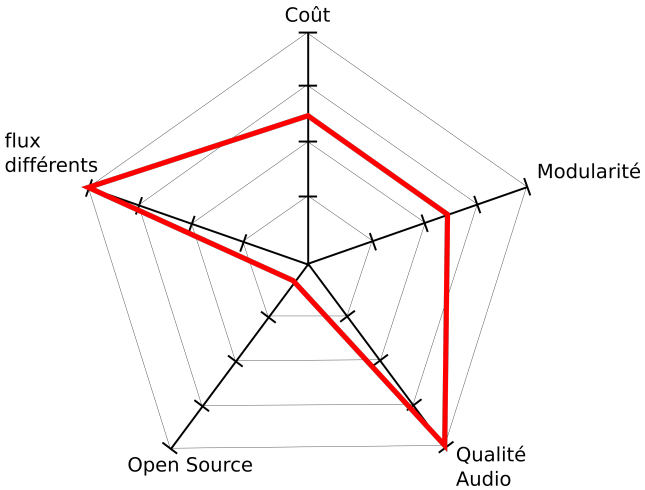
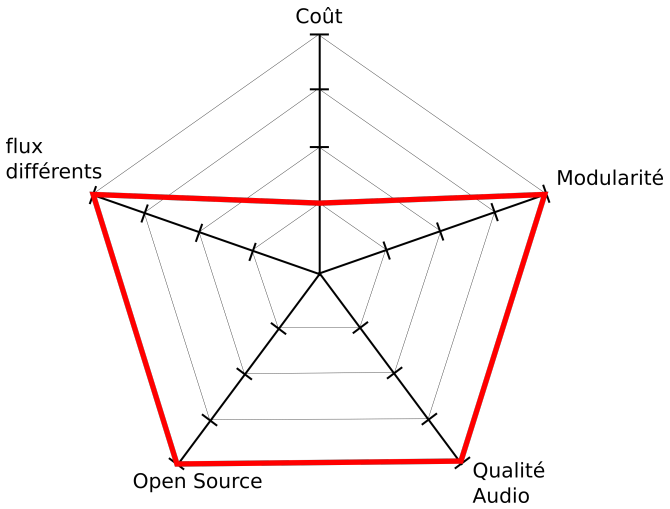



Figure 2 : Graphe radar pour une Google Chromecast Audio

Le but de notre projet sera alors de correspondre à un graphe radar de la forme suivante :



# 2 Méthode de travail

Dans un premier temps nous nous sommes  réunis avec nos tuteurs, Rodlphe ASTORI et Alexandre BOE afin de faire un point de départ sur le projet, ainsi que la manière dont nous allions nous y prendre.

Il nous a alors été proposé de travailler selon la méthode Scrum dont un des principes est de découper la durée totale du projet en plusieurs « sprints » d'une quinzaine de jours. Un objectif est fixé au début de chaque sprint et nous devons tenter de s'y tenir, en ne progressant que dans la direction donnée. A la fin de chaque sprint une réunion est organisée de manière à faire le point sur ce qui a été fait, définir les objectifs du prochain sprint et éventuellement corriger la trajectoire.

Les deux premières semaines de notre projet ont été consacrées aux recherches préliminaires et nous avons commencé le fonctionnement par sprints la semaine du 15 octobre 2015.

# 3 Architecture des modules

## 3.1. Présentation des modules

Afin de clarifier notre discours, nous appellerons « module » le haut-parleur et ses équipements embarqués, la partie applicative pour streamer la musique étant mise de côté.

Dans le cadre de ce projet, nous sommes partis sur deux types de modules distincts puisque nous souhaitons proposer au public une solution modulaire et non fermée. Dans les deux cas, nous avons choisi d'ajouter à la Raspberry Pi une carte son USB afin d'améliorer considérablement la qualité sonore que nous envoyons vers le haut-parleur.

## 3.2. Type de module avec HP actif

Le premier module embarque une Raspberry Pi pour la réception de la musique ainsi qu'une carte son USB. (voir Annexe)

Cette solution permet à l'utilisateur de connecter n'importe quel haut-parleur actif à la carte son avec un connecteur jack 3.6 mm. Cette solution permet un montage très rapide du dispositif pour une utilisation iminente.

## 3.3. Type de module avec HP passif

Le second module embarque également une Raspberry Pi ainsi qu'une carte son USB.

Cependant ce module embarque aussi un amplificateur audio auquel l'utilisateur pourra y connecter le haut-parleur passif de son choix. (voir Annexe)

L'avantage de cette solution est de maîtriser la qualité audio du système et de ne fournir qu'une seule « boîte » avec solution toute intégrée.

## 3.4. Chaîne de transmission

Le principe de la chaîne de transmission complète est le suivant : les différents modules (au nombre qu'on souhaite) sont connectés à une box domestique ou un routeur, où un Smartphone, une tablette ou un ordinateur est connecté. L'utilisateur peut ainsi streamer une ou plusieurs musiques vers les modules qu'il souhaite via ce réseau.


## 3.5. Une solution « Do It Yourself » pour des coûts maîtrisés

Le coût est une des données importantes de ce projet puisque nous avons vu précédemment qu'un haut-parleur multi-room du marché actuel est très onéreux. Il faut compter (après établissement d'un état de l'art) de 100 à 500 euros par haut-parleur. Il est donc évident que l'achat d'une enceinte par pièce revient rapidement cher.

Notre objectif sera alors d'obtenir un module ayant un coût maxium de 80 euros pour une solution prête à l'emploi. La modularité du projet permettra à l'utilisateur de mettre le prix qu'il souhaite pour la qualité sonore de son choix.

Le danger majeur porte sur la qualité audio qui diminue très rapidement avec le coût. La somme de 80 euros est un bon compromis pour notre projet.

Nous avons aussi vu très récemment que la Raspberry Pi Zero à 6 euros d'une puissance suffisante pour notre projet était disponible, contre une trentaine d'euros pour la solution actuelle. Elle a également l'avantage d'être beaucoup plus compacte que le modèle B+. Malheureusement, son succès est tel que sa disponibilité immédiate est quasi impossible.

# 4 Déroulement du projet

## 4.1. Recherches préliminaires

Les deux premières semaines ont été consacrées aux recherches préliminaires ainsi qu'à réfléchir quant aux solutions à mettre en place pour le mener à bien. Nous nous sommes convenus qu'il fallait développer :

- Une enceinte embarquant un amplificateur audio et un système pour recevoir le Wi-Fi
- Une application Smartphone permettant de choisir et streamer le flux musical
- Une application PC dotée des mêmes fonctionnalités

Pour le côté haut-parleur, nous avons choisi d'utiliser une Raspberry Pi disposant d'une distribution Linux embarquée et permettant ainsi de lui ajouter simplement un dongle Wi-Fi ainsi qu'une carte son USB. Cette solution nous est naturellement venue à l'esprit car elle est simple à mettre en oeuvre et relativement peu coûteuse.

Pour l'amplification, nous avions retenu deux solutions envisageables : un amplificateur à base de LM386 et un à base de LM3886.
Le premier est peu coûteux mais ne permet d'obtenir qu'une puissance de sortie faible et un son de qualité médiocre, tandis que le deuxième permet d'avoir une bonne puissance de sortie et un son de bonne qualité au détriment d'être un peu plus coûteux.

Pour l'application qui envoie la musique vers les haut-parleurs, nous nous sommes dans un premier temps intéressés à une application sur PC. Nous pensions développer l'application en C++ en utilisant Qt, un framework graphique fonctionnant sur toutes les plateformes (Windows, Linux et Mac OS).

Nous avons aussi réalisé le cahier des charges présenté en première partie.

L'aspect modulaire est défini par le fait de pouvoir inter changer facilement les différentes parties comme l'amplificateur, les haut-parleurs, la carte son,  etc. L'objectif à long terme étant un produit dont le coût est maîtrisé et qui correspond à chacun.

## 4.1. Sprint #1

### 4.1.1. Définition des objectifs

Comme décrit dans la section « Methode de Travail », nous avons défini les objectifs à atteindre en fin de sprint #1.
L'objectif était : **Envoyer un flux musical vers une Raspberry Pi, connectée à un haut-parleur**.

### 4.1.2. Application PC utilisant Qt et VLC

Nous avons alors pris la décision de partir sur une application destinée à fonctionner sur un PC. Nous avons choisi de développer cette application en C++, en utilisant le framework Qt, car il permet la réalisation de programmes graphiques sur toutes les plate-formes.

Une des voies envisagée était d'utiliser une librairie VLC intégrable à Qt, disponible sur le site **vlc-qt.tano.si**. L'avantage de passer par VLC pour streamer un flux musical est que c'est un logiciel Open Source, disponible à la fois sur Windows, Linux, Max OS et qui s'intègre parfaitement à Qt .

### 4.1.3. Résultats du Sprint

**Durant la réunion de ce Sprint**, nous somme parvenus à streamer un flux musical vers la Raspberry Pi seulement depuis Mac OS (avec AirPort).

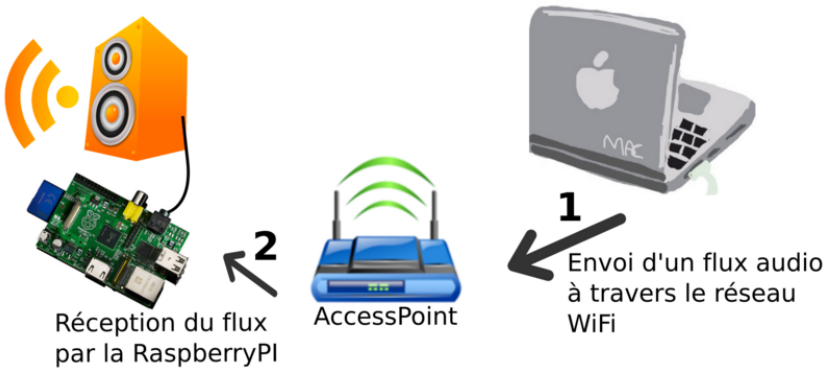



*Figure 9 : Démonstration du streaming depuis un Mac*

La démonstration était peu convaincante et assez restreinte par rapport au résultat attendu. Pour le sprint #2, nous avons alors déterminé qu'il était nécessaire de faire une veille technologique complète des solutions existantes, et de se concentrer sur l'application mobile dans un premier temps. En effet, l'intérêt principal de pouvoir streamer de la musique dans plusieurs pièces à la fois est de pouvoir contrôler la musique tout en se déplaçant. L'intérêt de contrôler le flux musical à l'aide d'un PC est donc assez limité.

## 4.2. Sprint #2

### 4.2.1. Définition des objectifs

Pour le Sprint #2, l'objectif principal était d'**envoyer cette fois deux flux audios différents vers deux haut-parleurs différents.**
En plus de ce principal objectif, il nous a été demandé de **réaliser une veille technologique** décrivant les solutions existantes autant au niveau logiciel et matériel DIY[1] que celles présentes sur le marché.

### 4.2.2. Veille technologique du point de vue matériel

Un large échantillon des solutions commercialisées (sous le nom d'enceintes multiroom) est présenté ci-après :

- Google Chromecast, à partir de 39€
- Samsung, à partir de 99€
- Bose, à partir de 199€
- LG, à partir de 179€
- Sonos, à partir de 229€
- Yamaha, à partir de 299€

Toutes ces solutions sont compatibles DLNA, permettent la lecture de différents flux audios sur différents haut-parleurs et sont livrées avec une application prête à l'emploi.

*Figure 10 : Enceintes multirooms du marché actuel*

[1] Do It Yourself

L'intérêt de la Google Chromecast par rapport aux haut-parleurs multirooms existant est qu'elle permet de transformer n'importe quel haut-parleur en système intelligent. Il suffit en effet de la connecter au haut-parleur ou à un amplificateur audio.

Par analogie à nos architectures de modules, la Google Chromecast remplacerait le système {Raspberry Pi + Dongle Wi-Fi + Carte Son USB).

*Figure 11 : Connexion de la Google Chromecast Audio*

En comparant ces deux systèmes (sans amplificateur ni haut-parleur) nous pouvons déjà constater une nette différence dans le prix :

| **Notre solution** | **Prix** |
|---|---|
| Raspberry Pi[2] | 6€ |
| Dongle Wi-Fi | 10€ |
| Carte son USB | 3€ |
| **TOTAL** | **19€** |
| | |
| **Solution du marché** | **Prix** |
| Google Chromecast Audio | **39€** |

[2] Sur la base de la Raspberry Pi Zéro

### 4.2.3. Veille technologique du point de vue logiciel

#### *Côté Serveur (application permettant le choix de la musique et sa direction)*

Nous avons également réalisé des recherches sur les logiciels existants (autre que les logiciels constructeurs des haut-parleurs précédemment présentés) permettant le streaming de flux audios. Ces recherches se sont portées sur les principaux systèmes d'exploitation, et plus particulièrement pour les Smartphones.

En effet, nous nous sommes convenus avec nos tuteurs qu'il était judicieux de privilégier en priorité l'application mobile plutôt qu'une application pour PC. Les solutions déjà existantes et qui semblaient être les plus performantes et les moins compliquées à mettre en place sont :

- Pour Smartphones :
  - Android : BubbleUPnP
  - iPhone : AirPlay (natif)



- Pour Ordinateur :
  - Windows : Windows Media Player (natif)
  - Max OS : AirPlay (natif et opérationnel avec iTunes)


#### *Côté Client (application au sein du haut-parleur, permettant de jouer la musique reçue)*


Nous avons choisi pour la Rapsberry Pi d'installer un OS Linux modifié nommé **Pi MusicBox**[3]. Ce système d'exploitation est un projet Open Source gérant les flux musicaux. Il intègre en effet un serveur/client DLNA, la possibilité d'y associer un compte Spotify ou encore de streamer un flux YouTube ou une musique située sur une clé USB connectée à la Raspberry Pi. Il dispose également d'une interface web permettant de contrôler facilement tous les flux. Son code est également **Open Source**. C'est donc la solution idéale pour streamer facilement de la musique depuis un mobile en utilisant le protocole DLNA.

[3] Plus d'informations sur http://www.pimusicbox.com/

#### 4.2.4. Résultats du Sprint

Lors de la réunion de fin de Sprint, nous avons effectué une démonstration d'une chaine complète. Nous sommes parvenus à diffuser deux musiques différentes depuis un même Smartphone vers deux haut-parleurs différents. (Voir Annexe)

L'objectif fixé en début de Sprint a donc été atteint et nous avons pu définir de nouveaux objectifs pour le sprint suivant :

- Réaliser la même chose mais avec notre propre application (BubbleUPnP n'étant pas Open Source)
- Réaliser la carte électronique du circuit d'amplification à base de LM3886

### 4.3. Sprint #3

#### 4.3.1. Définition des objectifs

Pour le Sprint #3, l'objectif principal était d'**envoyer deux flux audios différents vers deux haut-parleurs différents, mais avec notre propre application.**

Le second objectif consistait en la réalisation d'un amplificateur audio.

#### 4.3.2. Réalisation d'une application Android

Pour ce sprint, nous nous sommes concentrés sur la réalisation d'une application Android. En effet le système iOS (sur iPhone) dispose déjà d'une solution intégrée. Nous avons donc pris la décision de ne rien développer sous iOS.

Nous avons choisi de développer notre application sous **Android Studio**. Elle est donc réalisée en Java et nous avons fait en sorte qu'elle soit compatible avec tous les téléphones disposant de la version 4.1 (Jelly Bean) ou plus. Cela permet qu'un maximum d'utilisateurs puisse profiter de l'application.

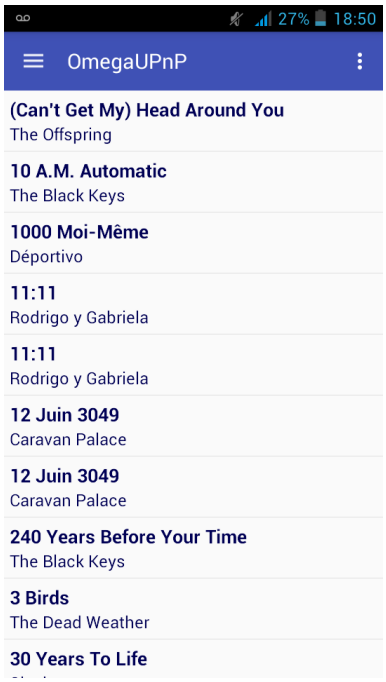



*Figure 13 : Première version du lecteur*

Nous avons dans un premier temps créé une **activity Android standard**, disposant d'un panneau latéral nous permettant par la suite de rajouter les fonctions de recherche de client DLNA (afin que la découverte des haut-parleurs connectés sur le même réseau se fasse automatiquement).

Dans un premier temps, nous avons créé une **classe Song** qui contient les variables liées à chaque chanson et les méthodes nécessaires pour pouvoir les manipuler.

Il a ensuite été nécessaire de créer une **classe MusicService**, contenant les méthodes permettant de manipuler les chansons. Elle permet de lier la liste des chansons au lecteur de média. Celui-ci est un service sous Android et est donc indispensable à lier si l'on veut lite un média.

Une fois ces étapes faites, nous avons affiché cette liste dans l'activité principale et l'avons trié par ordre alphabétique (*voir figure 13)*. On initialise alors le service permettant de lire les médias et on lit l'action de toucher une chanson dans la liste, au démarrage de ce service. On a donc une première version du lecteur musical, limitée à la lecture en local (sur le téléphone).

On a ensuite ajouté une basse permettant de contrôler la lecture (lecture, pause, piste suivante, etc.). Les contrôles étant déjà présents dans le service **Media Player**, il était seulement nécessaire de rajouter le widget contenant notre barre à l'activité principale, puis de lier les boutons aux contrôles. Cette amélioration est visible sur la figure ci-contre.

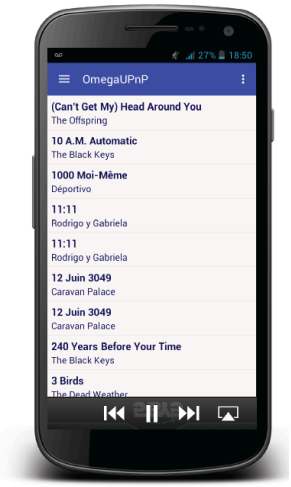



*Figure 14 : Ajout d'une barre de contrôle à notre application*

### 4.3.3. Réalisation de la carte d'amplification, côté module

Nous avons dans le même temps réalisé le layout de deux PCB[4] : un premier pour l'amplificateur du module avec haut-parleur passif (*voir section 3.3)*, un second pour l'alimentation de cet amplificateur.

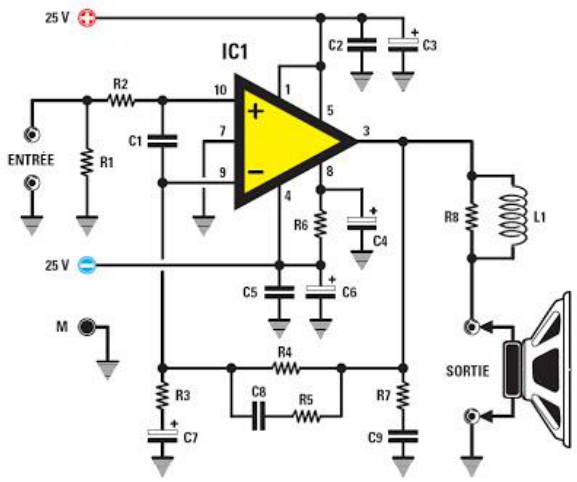



*Figure 15 : Schéma de base de l'amplificateur*

L'amplificateur a été réalisé à base de LM3886. Afin de dissiper la chaleur importante de ce composant, nous avons commandé un dissipateur thermique sur lequel le LM3886 sera vissé.

Disposant de peu d'expérience dans l'amplification audio et souhaitant une bonne qualité pour notre projet, nous nous sommes fortement insiprés de l'amplificateur ci-contre. De plus, les informations sur le site[5] correspondant sont très détaillées et nous permettent de comprendre l'importance de la plupart des composants choisis.

Par rapport au circuit d'origine, nous avons ajouté une LED de fonctionnement de l'amplificateur, un interrupteur ON/OFF ainsi qu'un potentiomètre rotatif d'une valeur de 50kΩ pour agir sur le volume sonore.

L'entrée audio est directement reliée à la sortie audio de la Raspberry Pi (sortie audio de la carte son connectée en USB.

Pour la sortie audio, nous avons prévu un bornier standard pour connexion de Haut-Parleur actif.

La version du PCB lors de ce sprint est disponible en Annexe.

*Figure 16 : Bornier de sortie audio*

[4] Printed Board Circuit (Carte électronique)
[5] http://www.schema-electronique.net/2011/12/un-amplificateur-final-hi-fi-de-40-70.html

#### 4.3.4. Résultats du Sprint

Les objctifs de ce sprint n'ont donc pas tout à fait été atteints :

- Le lecteur Android ne permet que de lire localement les musiques.
- Les PCB n'ont pas été gravés.

Il a donc été nécessaire pour la suite de corriger au plus vite ces deux points de manière à ce que le projet puisse aboutir.

## 4.4. Séparation des tâches

Alexandre JOUY et moi-même Julien HERIN nous sommes réparti naturemment les tâches puisque deux parties bien distinctes se sont montrées lourdes en réalisation.

Alexandre JOUY s'est donc concentré sur la partie serveur, à savoir l'application permettant l'envoi des musiques vers les enceintes souhaitées.

Je me suis de mon côté concentré sur la partie « Hardware[6] », à savoir la réception des musiques côté haut-parleur, et la réalisation de tout le circuit d'amplification ainsi que le design du contenant.

Au retour des congés de décembre, nous avons cessé de travailler par sprints afin de ne pas se donner d'objectifs particuliers pour une date particulière, mais plutôt se concentrer au maximum sur les tâches à réaliser sans contraintes.

[6] Matériel

## 4.5. Réalisation de la carte d'amplification

### 4.5.1. Design et assemblage de la carte d'amplification

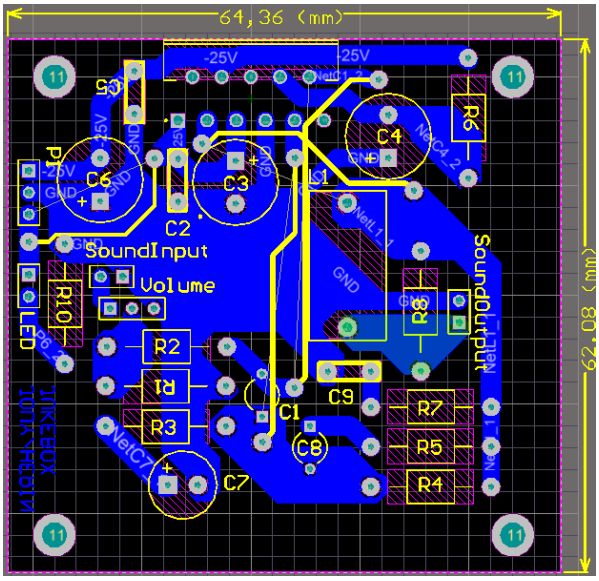
La carte d'amplification a été réalisée sous le logiciel Altium, déjà utilisé durant notre cursus IMA, mais aussi beaucoup utilisé lors d'un stage au sein de l'entreprise Melexis.

Nous avons beaucoup modifié la carte d'amplification à tirer par rapport à celle présentée lors du sprint #3.

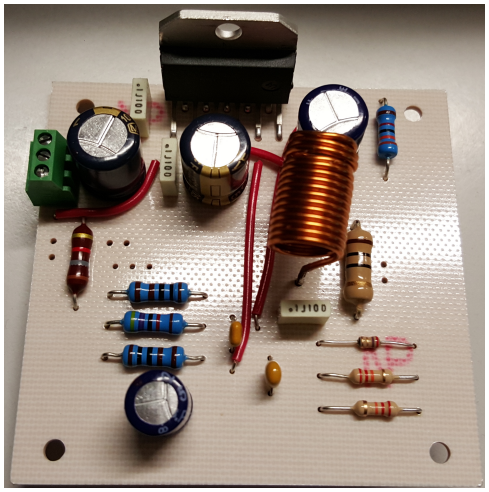
En effet, il était nécessaire de placer le composant LM3886 bien en bord de carte pour pouvoir le visser aisément au dissipateur. De plus, les circuits **masse, +25V et -25V** sont bien plus séparés.

Sur la première version, nous avions également mal dimensionné les composants. Ces derniers étaient donc très serrés.

La version finale de l'amplificateur est donc la suivante :




Une fois la carte assemblée, on peut remarquer sur la photo ci-dessous que la carte est bien plus présentable.



Sur la dernière version, nous avons aussi placé le potentiomètre de volume directement après l'entrée audio afin de ne pas influer sur le gain même de l'amplificateur LM3886. (*voir Annexe)*

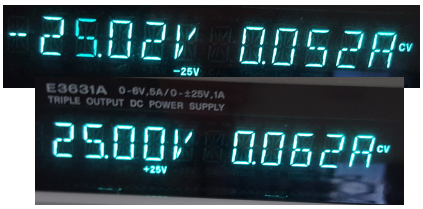
### 4.5.2. Contrôles, tests et caractérisation

## Contrôles et tests

Il a ensuite été nécessaire de contrôler et de caractériser le circuit d'amplification précédemment assemblé, afin d'en assurer le bon fonctionnement.

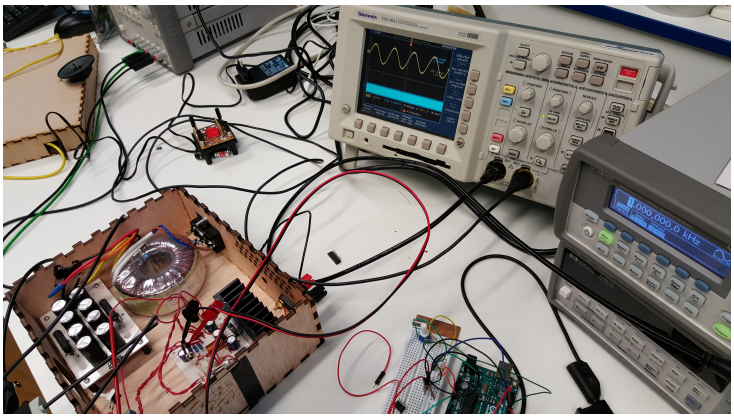
Nous avons tout d'abord contrôlé la consommation du circuit. Pour ce faire, nous avons connecté ce dernier à une alimentation de laboratoire disponible en salles de Travaux Pratiques. Nous avons ensuite limité le courant à 100 mA et alimenté le circuit avec une tension +/- 25V.

Si le circuit fonctionne correctement, la valeur limite de 100 mA en consommation ne peut être atteinte puisque le LM3886 ne consomme qu'environ 50 mA d'après la datasheet, le reste du circuit consommant très peu. Nous avons utilisé un haut-parleur passif de ¼ Watts pour une valeur ohmique de 22kΩ.
Sur l'alimentation de -25V, nous relevons une consommation d'environ **52mA** stable.
Sur l'alimentation de +25V, nous relevons une consommation d'environ **62mA** stable.




Le son sortant du haut-parleur est tout à fait audible. Mais cela ne suffit pas à caractériser notre amplificateur.

## Caractérisation



Nous avons donc remplacé le haut-parleur par un oscilloscope, et l'entrée audio (étant un téléphone portable) par un Générateur Basses Fréquences. De cette manière, nous pouvons aisément contrôler ce qui entre et ce qui sort de l'amplificateur.

Pour le signal du GBF, nous avons choisi quelques fréquences caractéristiques du spectre sonore. En ce qui concerne la tension, nous avons choisi une sinusoïde de 500 mV crête-à-crête, la datasheet nous signalant un maximum de 1V pour l'entrée audio du LM3886.

Voici les résultats de cette caractérisation :

| | **Tension de sortie** | | |
|---|---|---|---|
| **Fréquence** | Volume 25% | Volume 50% | Volume 100% |
| **N/A, pas d'entrée** | - | - | 9.20 mV (bruit) |
| | | | |
| **20 Hz** | 146 mV | 480 mV | 1.02 V |
| **500 Hz** | 130 mV | 510 mV | 1.12 V |
| **1 kHz (500mV)** | 130 mV | 572 mV | 1.19 V |
| **8 kHz** | 140 mV | 524 mV | 1.15 V |
| **20 kHz** | 150 mV | 500 mV | 1.08 V |

Les légères fluctuations visibles dans les données précédentes sont probablement dûes à la non répétabilité des tests, à savoir de nombreux paramètres variant (potentiomètre de volume pas exactement au même endroit, câbles deffectueux, etc.)

On remarque cependant que les valeurs de sorties ne sont pas influencées par la fréquence d'entrée. Le circuit réagit alors de la même manière sur toute la bande audible (20Hz-20kHz). En effet, la bobine placée sur le circuit maintient constante l'impédence de sortie sur toute la bande audible.

On remarque également un **bruit sur le signal de sortie** d'amplitude environ **9-10 mV** (voir Annexe).

Si on convertit le signal de sortie maximum obtenu précédemment et le bruit de sortie en dBm, on obtient (conversion réalisée pour une charge 600 Ohms pour du matériel audio) :

| **1.19 V (signal)** | **+4 dBm** |
|---|---|
| **10 mV (bruit)** | -38 dBm |

Le rapport Signal/Bruit étant égal à $P_{signal,dB} - P_{bruit,dB}$, on obtient un rapport signal/bruit d'environ **42 dB**.

La datasheet quant à elle indique un rapport signal/bruit de **92.5 dB** à 1kHz. Notre circuit autour du LM3886 ainsi que les signaux parasites ont donc ajouté du bruit.

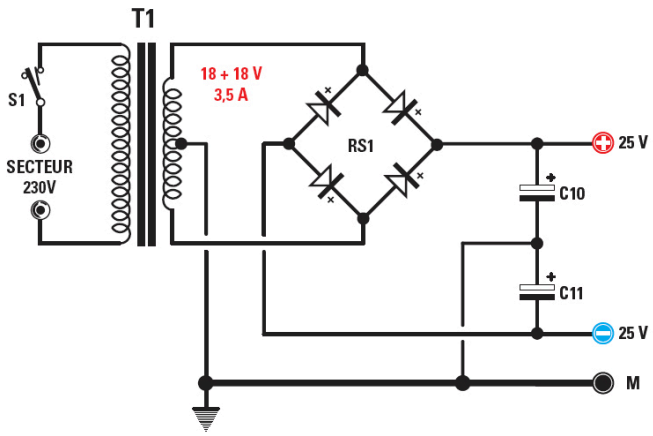
Nous aurions aussi pu mesurer le taux de distorsion harmonique. Nous avons cependant préféré utiliser ce temps pour d'autres tâches. La datasheet indique un THD de **0.03 %**.

## 4.6. Réalisation de la carte d'alimentation

Dans la version finale du projet, l'amplificateur doit être alimenté en +/-25V, comme nous l'avons fait pour les tests avec une alimentation de laboratoire. Mais cette fois-ci, cette alimentation devra être intégrée au module.

Nous nous sommes alors basés sur un schéma classique d'alimentation (pour passer d'un courant alternatif à un courant continu).

Le schéma ci-contre est celui fourni par le même site fournissant la carte d'amplification. Cependant afin d'obtenir une alimentation bien plus correcte, nous avons souhaité ajouter un filtre passe-base pour diminuer au maximum les fluctuations de tension en sortie.




Cette fois encore, nous nous sommes seulement basés sur le schéma donné, mais nous avons passé beaucoup de temps sur le choix final des composants utilisés sur cette carte.

Pour le transformateur, nous avons choisi la référence **VTX-146-120-118**. C'est un transformateur fournissant deux secondaires de **18V** chacun (tension efficace) avec un courant de **3.33 A maximum** (suffisant pour notre amplificateur).

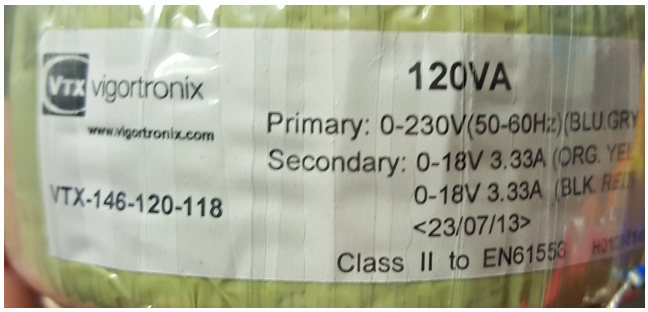
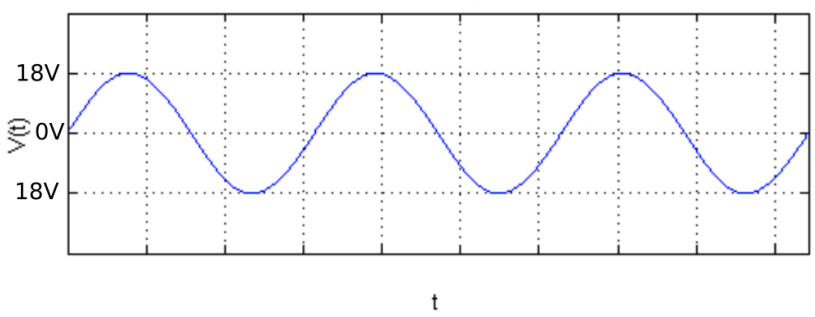


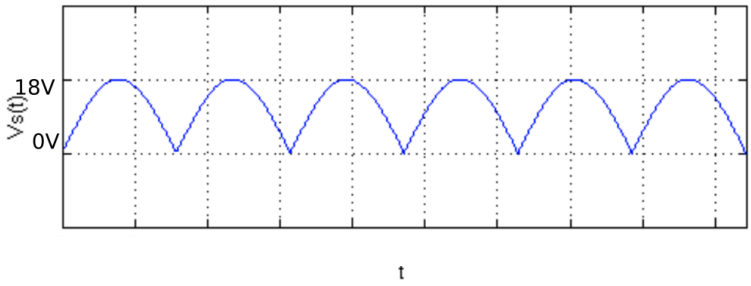

*Figure 24 : Caractéristiques du transformateur choisi*

### 4.6.1. Principe de fonctionnement attendu

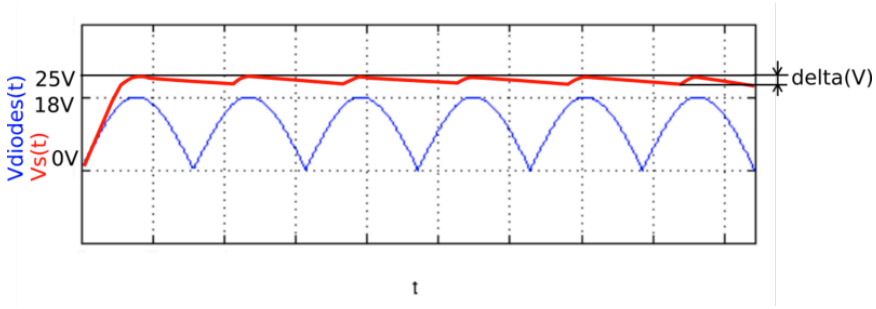
Le signal fourni par les bobines secondaires du transformateur 220V/18V est le suivant :




Après double redressement par le pont de Graetz, on se retrouve avec le signal suivant :



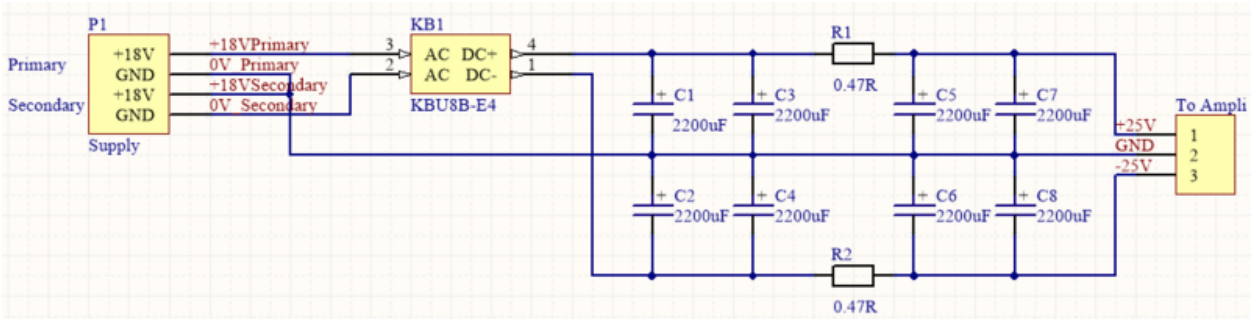

Après filtrage par les condensateurs, on se retrouve avec un signal élevé et lissé :




La différence entre les tensions maximum et minimum de la tension de sortie (delta(V)) est appelée tension de « ronflement ». Notre but est donc de diminuer cet écart afin d'obtenir une alimentation la plus stable possible pour notre amplificateur, au risque de déteriorer le matériel.

### 4.6.2. Dimensionnement du filtre

Pour pallier à ces fluctuations, nous avons ajouté un filtre CRC au montage, visible sur le schéma suivant (schéma final de notre alimentation) :




NB : le composant nommé « KB1 » est notre pont de Graetz.

Nous voulons en effet réaliser un filtre passe-base à environ 100Hz (fréquence relative au double redressement de la fréquence 50Hz).

Comme la fréquence de coupure d'un filtre RC est donnée par la formule : 1/(2*Pi*R*C)
avec C la valeur de la capacité et R la valeur de la résistance, on obtient avec les valeurs du schéma une fréquence de coupure d'environ **102 Hz**. Ces calculs sont valables pour un filtre passe-bas du premier ordre, mais sont adaptables à notre cas.

### 4.6.3. Simulations sous Altium Designer

Nous avons quand même préféré effectuer des simulations sous Altium Designer.

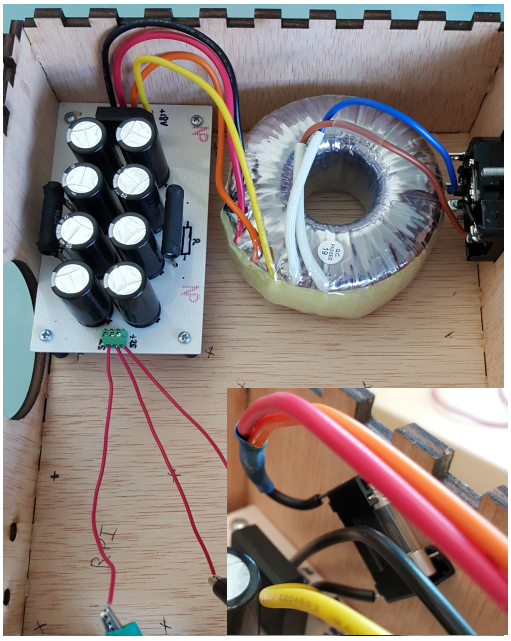
On ne se basera que sur les composants de la moitié supérieure (C1, C3, R1, C5 et C7) car les résultats seront les mêmes pour la moitié inférieur (il s'agit d'une alimentation symétrique).

- Nous avons cherché à determiner l'influence de la **valeur capacitive d'entrée du filtre** (C1+C3) : voir Annexe. On remarque que plus elle est élevée, plus la tension efficace de sortie sera élevée.

- Ensuite, nous nous sommes penchés sur la **valeur de la résistance** (R1) : voir Annexe. On voit que plus elle est faible, plus la tension de sortie est importante, mais on augmente les fluctuations.

- Pour finir, nous avons cherché à déterminer l'influence de la valeur capacitive de sortie du filtre (C5+C7) : voir Annexe. Plus elle est élevée, plus la tension de sortie est lissée. En effet, les variations de tension sont bien mieux contrées.

### 4.6.4. Assemblage et tests



La carte d'alimentation a finalement été gravée, puis assemblée.

On peut voir sur l'image ci-contre le montage d'alimentation avec le connecteur IEC (muni d'un fusible 3.15 A et d'un interrupteur), le transformateur ainsi que la carte d'alimentation. Sur la petite photo dans le coin, on a ajouté un fusible après les bobines secondaires afin de protéger notre circuit.

Monsieur Xavier REDON a testé notre alimentation pour des raison de sûreté et de sécurité. **Elle est bien fonctionnelle** avec une tension mesurée en sortie de **+27V/-27V**. Cette valeur est légèrement plus grande à celle attendue, ceci étant probablement dû au fait qu'il n'y avait **pas de charge** après le circuit.

## 4.7. Réalisation du boîtier du module avec amplificateur

Nous avons réalisé le boîtier du module avec amplificateur en parallèle du circuit d'alimentation. En effet, nous souhaitions tester le tout assemblé.

### 4.7.1. Description du boîtier

Ce boîtier est prévu d'être utilisé dans la version « module avec amplificateur » afin de contenir tous les éléments nécessaires en un unique bloc, à savoir :

- L'alimentation de l'amplificateur
- L'amplificateur
- La Raspberry Pi

Sur les parois de la boîte seront ajoutés :

- Des LEDs (fonctionnement, réception de flux audio disponible, etc.)
- Des boutons (ON/OFF pour l'amplificateur ainsi que pour la Raspberry Pi)
- Un bornier pour y connecter un haut-parleur passif externe (8ohms/50 watts maximum)

### 4.7.2. Réalisation du boîtier

Nous avons choisi de réaliser le boîtier du module en bois. En effet, cela nous permet d'utiliser la découpeuse LASER du Fabricarium, et le rendu final est très esthétique.

Il nous a suffi de nous rendre sur un site de génération de boîtes. Une fois le fichier sauvegardé (format .svg), nous y avons ajouté différents trous de manières à y intégrer tous les éléments décrits en section 4.7.1. De nombreux trous nécessaires à l'aération suffisante du boîtier y ont aussi été ajouté. (voir Annexe)

Toutes les instructions d'utilisation de la découpeuse LASER du Fabricarium sont disponibles sur www.fabricarium.fr.

### 4.7.3. Aération du boîtier

Le boîtier doit être convenablement aéré afin de dissiper la chaleur des composants électroniques qu'il contient, mais surtout celle du circuit d'amplification.

On voit sur l'image ci- contre que le dissipateur du circuit d'amplification se trouve proche des trous d'aération afin d'extraire au maximum la chaleur qu'il émet. Nous avons aussi ajouté des trous d'aération sur le dessus de la boîte puisque par convexion naturelle, la chaleur sera mieux évacuée.

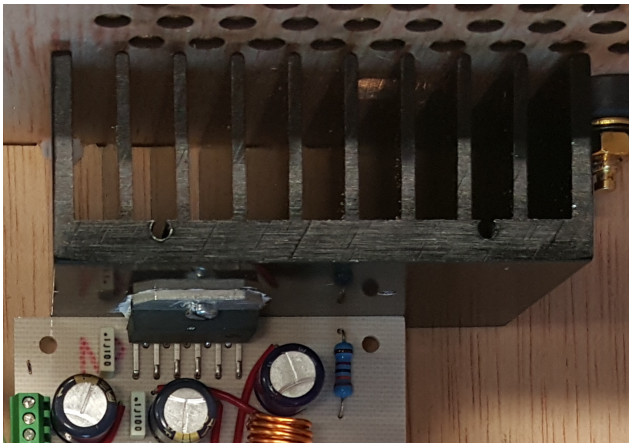


*Figure 34 : Dissipateur pour le circuit d'amplification*

Nous aurions bien entendu préféré ce dissipateur à l'extérieur du boîtier. Cependant, d'après la datasheet du LM3886, le pad vissé au dissipateur est au potentiel d'alimentation négative (-25V). Il aurait été préférable que celui-ci soit connecté à la masse.

De ce fait, il n'est pas possible de laisser à l'utilisateur la possibilité de toucher le dissipateur. En effet, même si son revêtement anodisé a une caractéristique isolante, il n'est pas à l'abri d'une rayure ou d'une oxydation avec le temps, le rendant conducteur.

Il aurait donc fallu utiliser des matériaux isolants entre le pad du LM3886 et le dissipateur, mais là encore, une oxydation peut avoir lieu avec le temps rendant le dissipateur conducteur.

En conclusion, nous avons choisi de sur-aérer le boîtier et d'empêcher tout risque d'électrisation.

## 4.8. Réalisation d'un circuit d'extinction propre de la Raspberry Pi

### 4.8.1. Description

La Raspberry Pi étant à l'intérieur de la boîte précédemment présentée et une connexion SSH[7] étant non prévue, il a été nécessaire de faire en sorte que celle-ci puisse être éteinte de manière correcte.

En effet, l'extinction « brutale[8] » pourrait très facilement corrompre la carte SD présente sur la Raspberry Pi et ainsi détruire toutes les configurations préalablement effectuées.

### 4.8.2. Réalisation du circuit

Pour la réalisation de ce circuit, nous avons pensé à un circuit à base de condensateurs et de relais.

Sur le schéma (disponible en Annexe), deux parties sont distinctes :

- La partie **non encadrée en rouge** est le circuit ON/OFF « normal », comprenant les boutons sur lesquels l'utilisateur agira.
- La partie **encadrée en rouge** est le circuit de sécurité (qui passe la Raspberry Pi sur batterie et qui lui donne la consigne de s'éteindre)

[7] Connexion à distance « Secure Shell »
[8] Débranchement de la prise d'alimentation principale par l'utilisateur ou coupure de courant

### 4.8.3. Principe de fonctionnement du circuit ON/OFF « normal »

Ce circuit sera placé entre la source de courant et la Raspberry Pi (au milieu d'un câble USB coupé en 2)

Le principe de fonctionnement du circuit ON/OFF « normal » est le suivant :

1. Quand le relai (REL1) est dans son état inactif (sur 1), le 5V arrivant de la source de courant va charger le condensateur (C1).

2. Quand le condensateur sera assez chargé, il activera les transistors 1 et 2 qui connecteront la bobine du relai (REL1) à la masse.

3. Quand on appuiera sur le bouton ON (ON_BUTTON), le relai passera de 1 à 10 (dans son autre état), la RPi sera donc alimentée. Cependant, le condensateur se décharge ! Et donc les transistors seront désactivés et le relai aussi --> La RPi ne sera donc plus alimentée.
   Pour éviter cela, on active une sortie de la RPi (Pin 3 de "To GPIO) pour fournir du 3.3V, et donc maintenir le condensateur chargé, donc le relai actif sur la pin 10. La RPi reste donc bien alimentée.

4. Pour l'extinction, on ajoute un retour (pin 6 de "To GPIO") qui va envoyer la valeur 1 lorsqu'on presse le bouton OFF (OFF_BUTTON). La LED (OFF_LED) s'allumera pour signaler à l'utilisateur que la RPi est en train de s'éteindre. Un script s'exécutera en même temps interrompant les programmes proprement, puis stoppant le 3.3V.

5. Le condensateur se décharge donc, puis au bout d'environ 1 minute (temps réglé en fonction de la valeur du condensateur (C1)), les transistors sont désactivés, le relai (REL1) le sera donc aussi (en position 1), la RPi ne sera plus alimentée.

6. On revient au début du cycle, le condensateur va se recharger, il ne restera plus qu'à presser le bouton ON (ON_BUTTON) quand on souhaitera allumer de nouveau la RPi.

### 4.8.4. Principe de fonctionnement du circuit de sécurité

Un boîtier de piles 6V est connecté à ce circuit. Un comparateur allume une LED en façade du boîtier pour prévenir si les piles doivent être changées. Puisque la Raspberry Pi doit être alimentée en 5V, nous avons ajouté un régulateur de tension de 5V.

Le principe de fonctionnement du circuit de sécurité est le suivant :

1. La bobine du relai (REL2) en haut du schéma est directement alimentée par le 5V de la source de courant. Lorsque ce relai (REL2) est alimenté, il est dans l'état 10, non connecté. Dans cette situation, la Raspberry fonctionne correctement avec la source d'alimentation normale et les piles ne sont pas utilisées.

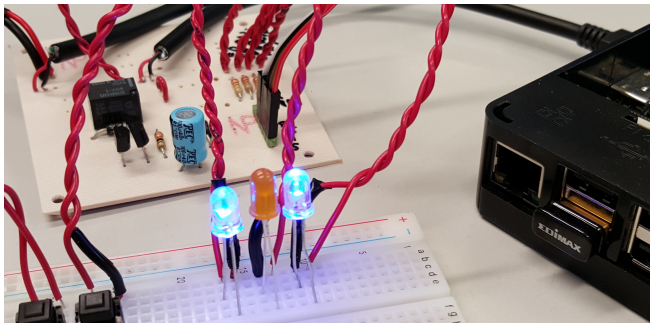
2. Si le courant principal est coupé, le relai (REL2) passera dans l'autre état (état 1) pour laisser les piles alimenter à leur tour la Raspberry Pi.

3. Un signal est aussitôt envoyé à la Raspberry Pi (pin 2 de « To GPIO ») pour lancer le script d'extinction de celle-ci.

4. Lorsque la Raspberry Pi sera éteinte, elle ne fournira plus de courant au condensateur (C2), qui au bout de quelques secondes désactivera le transistor (T3). Par conséquent, la Raspberry ne sera plus connectée à la masse, ce qui préservera les piles en stoppant leur fourniture de courant inutile.

### 4.8.5. Assemblage et tests

Le PCB réalisé sous Altium est visible en annexe. Avant de le valider, il a d'abord fallu le tester sur board de test, élément par élément. Une fois tous les tests concluant, nous avons assemblé le PCB final puis testé celui-ci



Le circuit d'extinction et d'allumage manuel seul de la Raspberry Pi fonctionne Pour le circuit de sécurité, le courant de collecteur du transistor mettant la Raspberry à la masse n'est pas assez élevé (34 mA mesuré à l'Ampère-mètre). Il aurait donc fallu diminuer la résistance de base de ce transistor, celle juste après le 3V3 de la Raspberry Pi et augmenter considérablement la valeur du condensateur. Par manque de temps, nous n'avons malheureusement pas pu pousser les tests.

## 4.9. Travail réalisé sur la Raspberry Pi

### 4.9.1. Dévérouillage des GPIO

La Raspberry Pi étant à l'intérieur du boîtier, il n'est pas évident de savoir si elle est bien en fonctionnement et inversement si elle est bien éteinte. Pour permettre à l'utilisateur d'avoir un retour clair sur l'état de la Raspberry Pi, nous avons pris la décision d'utiliser des LED montées sur les parois du boîtier du module.

Afin d'agir sur l'état des LED, nous avons fait des premiers tests avec la Raspberry Pi à l'aide de scripts Python. En effet, ce langage est très efficace quand il s'agit d'agir sur les GPIO de la Raspberry Pi puisque les scripts sont relativement courts. Cependant, les premiers tests étaient des échecs puisque la Raspberry Pi ne nous permettait pas d'agir sur les GPIO.

Après de nombreuses recherches, nous sommes tombés sur un forum où des personnes utilisant MusicBox (que nous avons également installé) rencontraient le même problème. L'explication était simple : MusicBox, dans sa configuration standard, est prévu pour travailler avec la carte son externe **HifiBerry DAC+** connectée sur les GPIO de la Raspberry Pi (figure ci-contre).

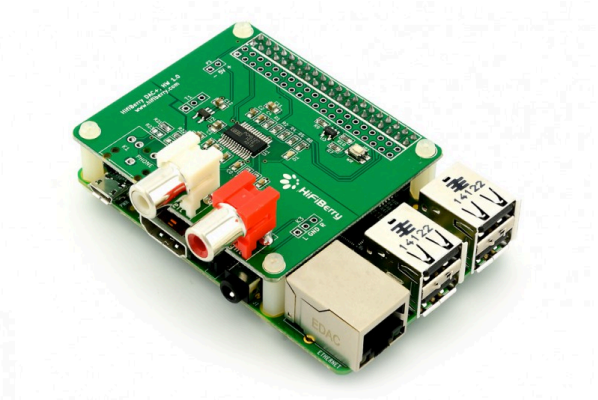


*Figure 39 : HifiBerry DAC+ connecté sur une Raspberry Pi 2*

La solution a donc été de commenter une ligne dans un fichier de configuration (/boot/config.txt), qui mettait les GPIO en configuration pour être utilisés avec cette carte son. Une fois ceci fait, il nous était très facile d'agir sur les GPIO.

### 4.9.2. Script d'extinction de la Raspberry Pi

Afin d'éteindre la Raspberry Pi sans utiliser de connexion SSH, nous avons installé un bouton externe. Nous avons ensuite écrit un programme en Python (disponible en Annexe) qui travaille **par interruptions** sur le bouton précédemment installé.

On définit la fonction qui sera appelée lors de l'interruption sur le bouton :

```
def shutdown(pini):          #La commande 'halt' est appelée et la Raspberry s'eteint
    call(['/sbin/halt'], shell=False)
    blink_off_led()
```

avec la fonction :

```
def blink_off_led():         #Une LED va clignoter pendant l'extinction
    while True:
        gpio.output(SWITCHING_OFF_LED, 1)
        time.sleep(0.3)
        gpio.output(SWITCHING_OFF_LED, 0)
        time.sleep(0.5)
```

Puis on ajoute un événement qui va intercepter l'interruption sur le bouton :

```
gpio.add_event_detect(SWITCH_OFF_BUTTON, gpio.RISING, callback=shutdown, bouncetime=200)
```

#### *Utilisation de Crontab & Screen*

Afin de lancer le script au démarrage de la Raspberry Pi, nous avons choisi d'utiliser Crontab et GNU Screen.

**Screen** permet de pouvoir lancer un programme dans une autre terminal que le terminal courant puis de le reprendre au besoin. Il nous permet dans notre cas de lancer notre script précédemment écrit en tâche de fond, sans perturber le bon fonctionnement de la Raspberry Pi et de MusicBox.

**Crontab** permet d'exécuter automatiquement des scripts ou des commandes au démarrage de la Raspberry Pi.

Nous entrons alors la commande « crontab –e »
A la fin du fichier, nous ajoutons la ligne « @reboot screen –d –m /home/switch/switch.py » afin de lancer notre script au démarrage.

A l'aide d'une connexion sur port série, nous avons vérifié que la Raspberry s'éteint bien correctement :

```
[  234.035924] watchdog stopped
[  241.709544] watchdog stopped
[  248.489807] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[  250.606730] reboot: Power down
```

### 4.9.3. LED de fonctionnement et de réception prête

Afin d'avertir l'utilisateur que la Raspberry est bien allumée et que la réception de musique est prête, nous avons décidé d'ajouter deux LED correspondant à ces deux états.

**Pour la LED de fonctionnement**, celle-ci est tout simplement connectée au 3V3 de la RaspberryPi. C'est aussi cette sortie qui alimentera le condensateur du circuit de fonctionnement « normal » présenté dans la section 4.8.

**Pour la LED de réception prête**, nous avons modifié le fichier /opt/musicbox/startup/sh qui se lance à l'initialisation de MusicBox quand la RaspberryPi démarre. Dans ce fichier, nous avons ajouté à la ligne 236 les deux lignes suivantes :

```
# Allume la LED quand la réception est prête
/usr/bin/python /home/switch/turn_wifi_led_on.py
```

La ligne 236 vient en effet juste après que la Raspberry ait été connectée au réseau Wi-Fi et que le processus d'initialisation de la réception de musique ait été réalisé.

#### 4.9.4. Création de la nouvelle image .iso

Dans le cadre du tutoriel « Do It Yourself » réalisé pendant ce projet (expliqué dans la section 7), nous avons pensé qu'il était intéressant de coper l'image de la carte SD une fois toutes les modifications précédentes faites. Cela permettra en effet à l'utilisateur de ne pas avoir à réaliser de nouveau ces modifications.

# 5 Coût du projet

Un des objectifs importants du projet est un **coût maîtrisé**.

## 5.1. Module sans alimentation ni amplificateur

Nous avons déjà vu en section 4.2.2 que pour une configuration « Google Chromecast Audio » (sans amplificateur ni alimentation), le coût de notre projet s'élève à une vingtaine d'euros contre une quarantaine pour la solution commerciale sus-citée.

## 5.2. Module avec alimentation et amplificateur

Pour le module avec alimentation et amplificateur, nous avons repertorié le matériel utilisé lors de notre projet et avons établit les coût.

Pour l'**amplificateur** que nous proposons, le prix est de **31,72 euros**.
Pour l'**alimentation** que nous proposons, le prix est de **72, 74 euros.**
Si l'on ajoute à cela le circuit d'alimentation de la Raspberry Pi par boutons, il faut compter une quinzaine d'euros.

On remarque que ce module se situe aux alentours de la **centaine d'euros**, correspondant à l'entrée de gamme des haut-parleurs connectés du commerce, contre quatre-vingt euros annoncés en début de projet. En revanche, l'utilisateur pourra toujours y ajouter l'amplificateur qu'il souhaite ainsi que le haut-parleur de son choix.

Concernant l'alimentation, le transformateur compte pour **61%** du coût total de celle-ci. Il est donc encore possible de réduire ce coût en trouvant un transformateur bas prix.

Au coût total de ce projet, il faut aussi ajouter la Raspberry (ajout négligeable si l'on utilise la Raspberry Pi Zéro à 6 euros).

# 6 Modularité du projet

La modularité est également un point essentiel de notre projet. En effet, un des objectif que chacun crée son propre haut-parleur multi-room est qu'il puisse modifier les éléments du boîtier à souhait.

Nous avons donc choisi (voir figure en Annexe) de séparer le boîtier en deux zones distinctes. La première zone à gauche de l'image est reservée à l'alimentation du module (Amplifiicateur + RaspberryPi) et est complètement protégée du reste. L'utilisateur y a seulement accès en retirant le capot de cette zone maintenu par des vis.

Le seconde zone quant à elle est la zone de modularité. C'est ici que l'utilisateur pourra y placer son propre circuit d'amplification, pourvu qu'il soit alimenté en 0/+25V ou bien -25/+25V. Autrement, il devra modifier également le circuit d'alimentation. Il pourra aussi utiliser la Raspberry de son choix.

Afin de rendre cette zone la plus modulaire possible, nous avons limité les soudures et les fixations par vis des éléments. C'est pourquoi nous avons choisi d'utiliser des connecteurs pour que les éléments soient rapidement connectés et fonctionnels. L'utilisateur aura aussi le choix d'utiliser un haut-parleur interne, externe, ou bien les deux.

# 7 Réalisation d'un tutoriel « Do It Yourself »

Nous avons pensé dès le début de ce projet qu'il était intéressant que celui-ci devienne communautaire et donc complètement « Do It Yourself ». En effet, un des objectifs étant de réduire au maximum les coûts, nous avons souhaité pleinement partager notre travail.

Afin que les instructions de ce projet soient transmises correctement, nous avons choisi de réaliser un tutoriel PDF, de type www.instructables.com. Ce PDF sera disponible ainsi que les fichiers nécessaires au projet (fichiers PCB, fichier SVG pour la boîte, etc.) sur notre page de projet IMA.

# 8 Améliorations possibles

De nombreuses améliorations sont envisageables pour continuer à mener à bien ce projet et le faire évoluer de manière innovante.

D'abord, **l'application Android** n'a pas été réalisée. Une application pour PC a été réalisée à la place. En effet, les instructions concernant le streaming en DLNA ne sont accesibles qu'avec des logiciels obsolètes depuis juin 2015. De cette date à aujourd'hui, rares sont les personnes développant sous Android pour ce genre d'application. De ce fait, il est encore très difficile de réaliser ce point, et il serait très intéressant qu'une application pour notre projet soit un jour développée.

Nous avons aussi pensé à un **ajustement du volume sonore** des haut-parleurs en fonction de là où l'utilisateur se trouve. Cela pourrait se faire soit de manière **statique**, en ajustant manuellement le volume de chaque haut-parleur puis en sauvegardant la configuration, soit de manière dynamique avec un système de Kinect au mur.

# Bilan

Ce projet nous a confronté à la difficulté de tenir un planning prévisionnel et de définir en collaboration avec le client les points à développer. Il a aussi fallu trouver les solutions les plus crédibles à la réalisation du projet.

Avec notre expérience en Android quasiment inexistante, il a été très difficile de mener à bien ce projet puisque c'était un des principaux objectifs. Heureusement la solution pour PC nous a sauvé et l'application est fonctionnelle en même temps d'être Open Source. Nous avons beaucoup appris, tant dans la gestion de projet, qu'en réalisation d'amplificateur et d'alimentation, chose que nous n'avions jamais faite auparavent.

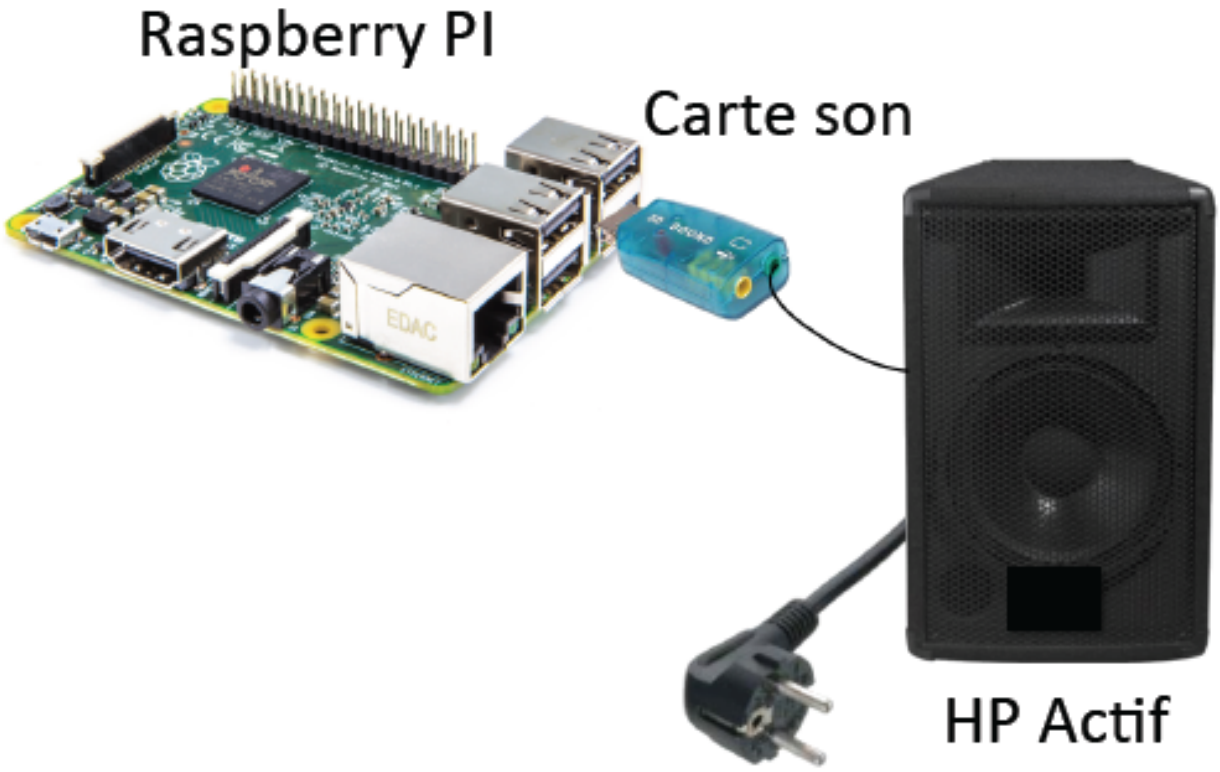
Nous aurions réellement souhaité apporté les améliorations citées précédemment pour être pleinement satisfaits du résultat, mais nous manquions de temps et d'organisation.

Aussi, l'impression de ne pas avancer et de bloquer sur des problèmes simples a été un grand défi que nous avons dû surmonter. Nous ne sommes pas pleinement satisfaits du prototype que nous livrons, mais au final le cahier des charges est en grande partie respecté.

Nous tirons de ce projet de fin d'Etudes une très bonne expérience de travail, de gestion du temps, du stress et de l'importance de la communication avec les collaborateurs et les clients.

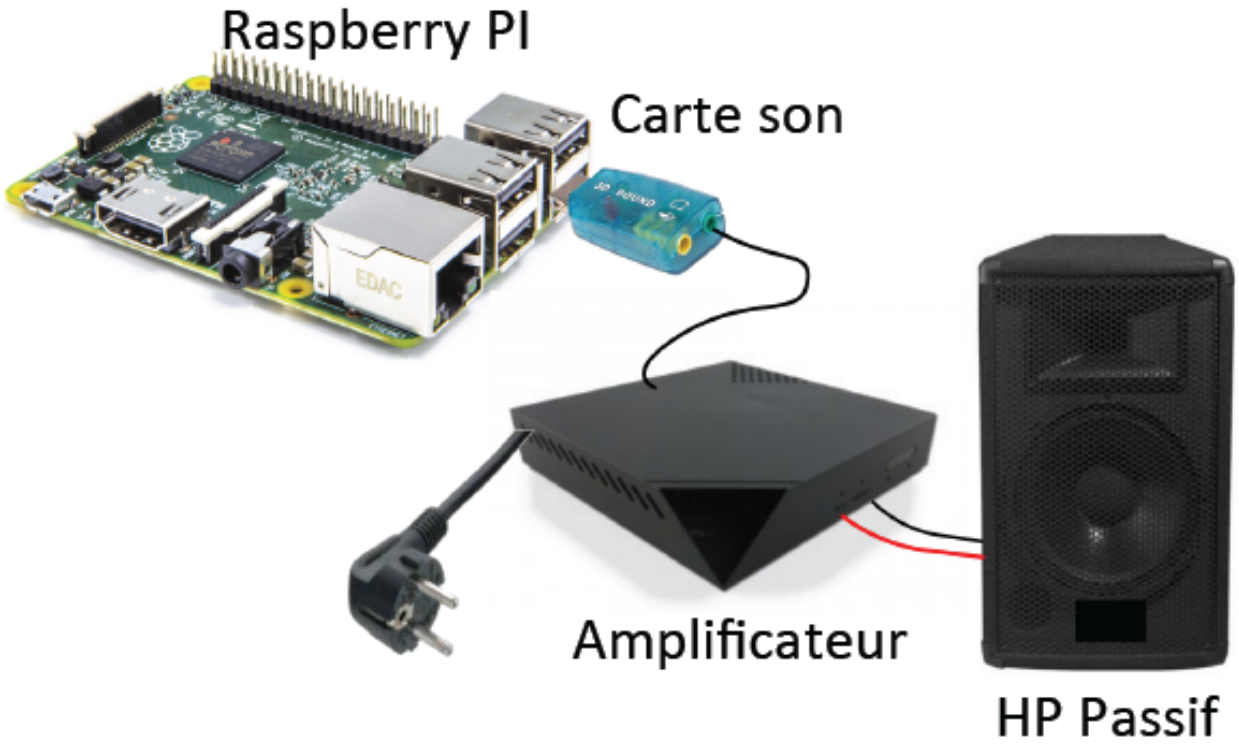
## ANNEXE 1

Module avec Haut-Parleur actif



## ANNEXE 2

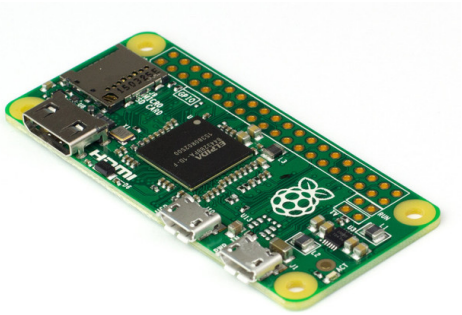
Module avec Haut-Parleur passif



# ANNEXE 3

## Comparatif Raspberry Pi B+ / Raspberry Pi Zéro

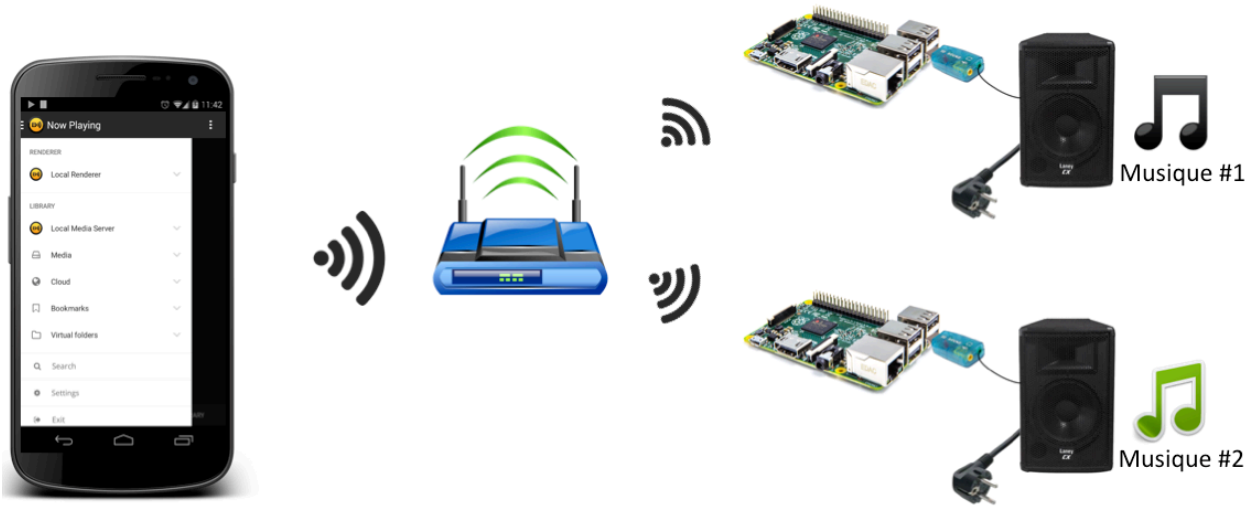
Raspberry Pi B+ : 35 euros



Raspberry Pi Zéro : 6 euros

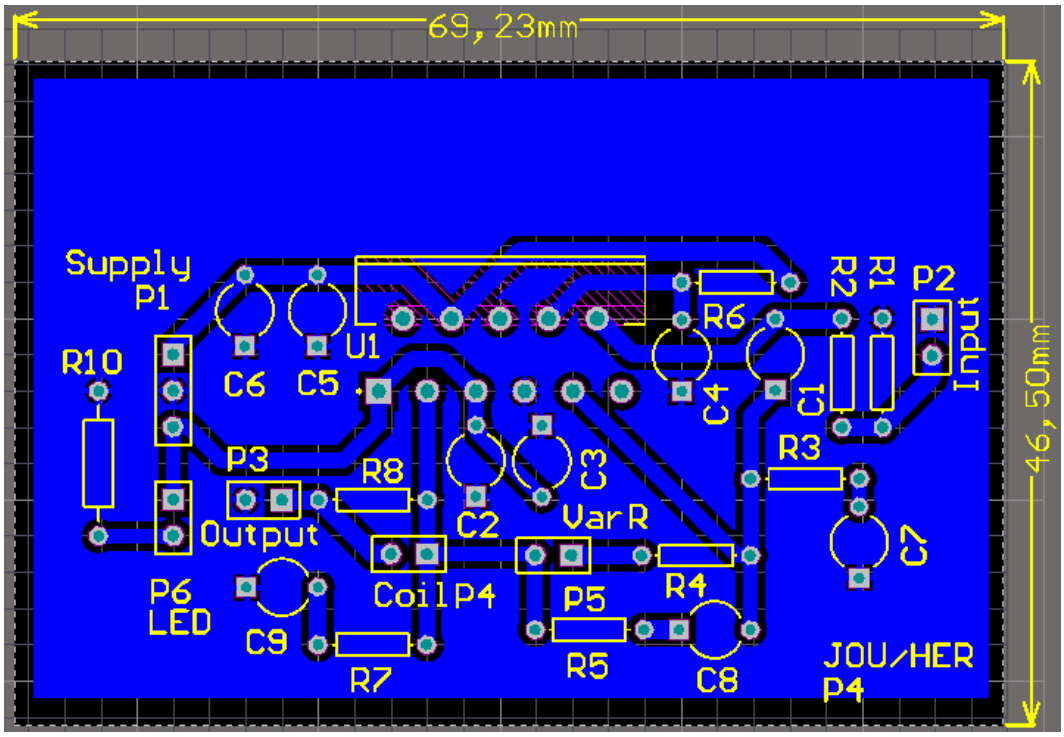
# ANNEXE 4

## Envoi de deux musiques vers deux haut-parleurs



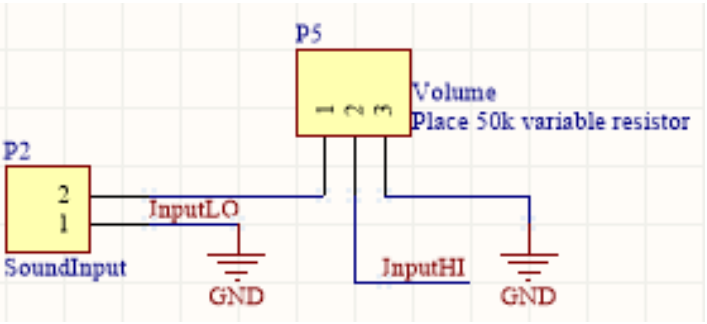
## ANNEXE 5

Première version du PCB d'Amplification audio



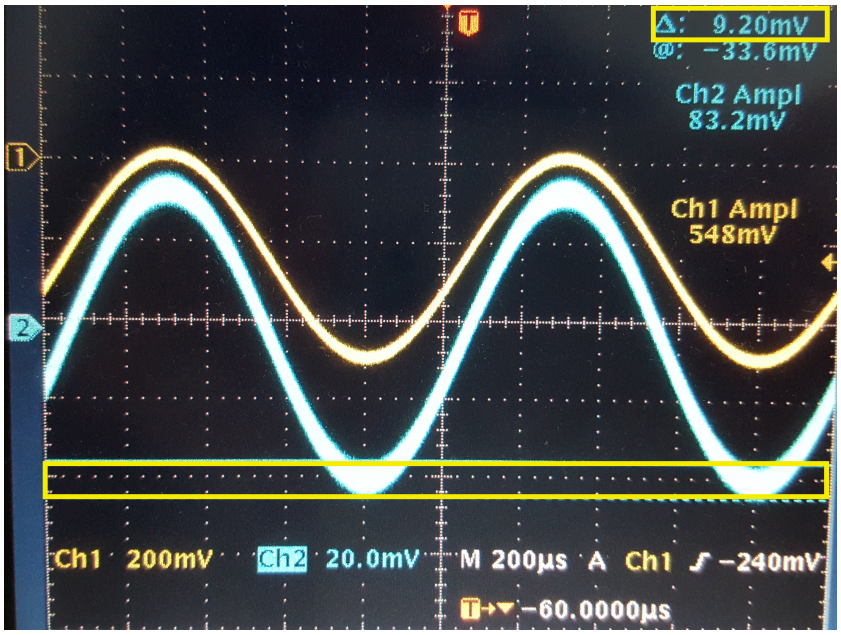
## ANNEXE 6

Contrôle du volume en entrée de l'amplificateur audio



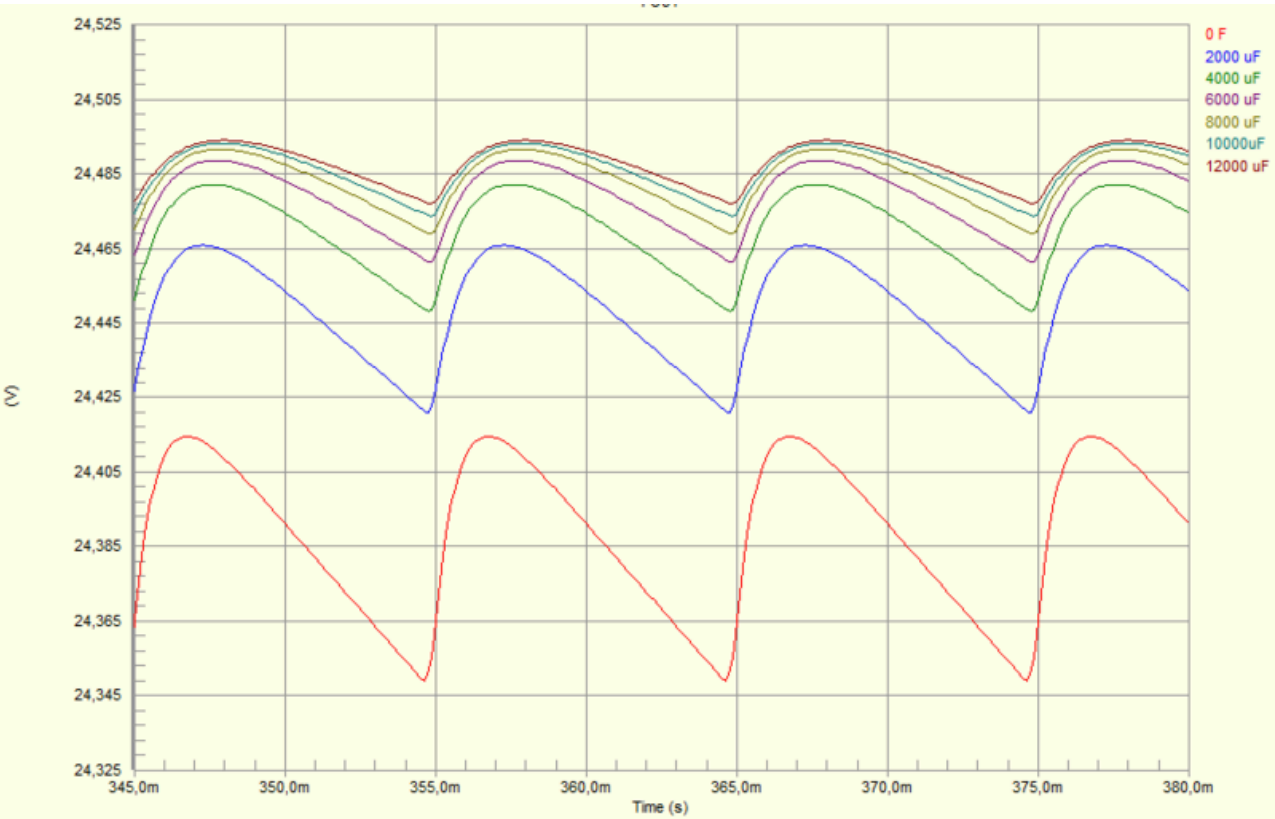

## ANNEXE 7

Détermination du rapport Signal sur Bruit de l'amplificateur audio



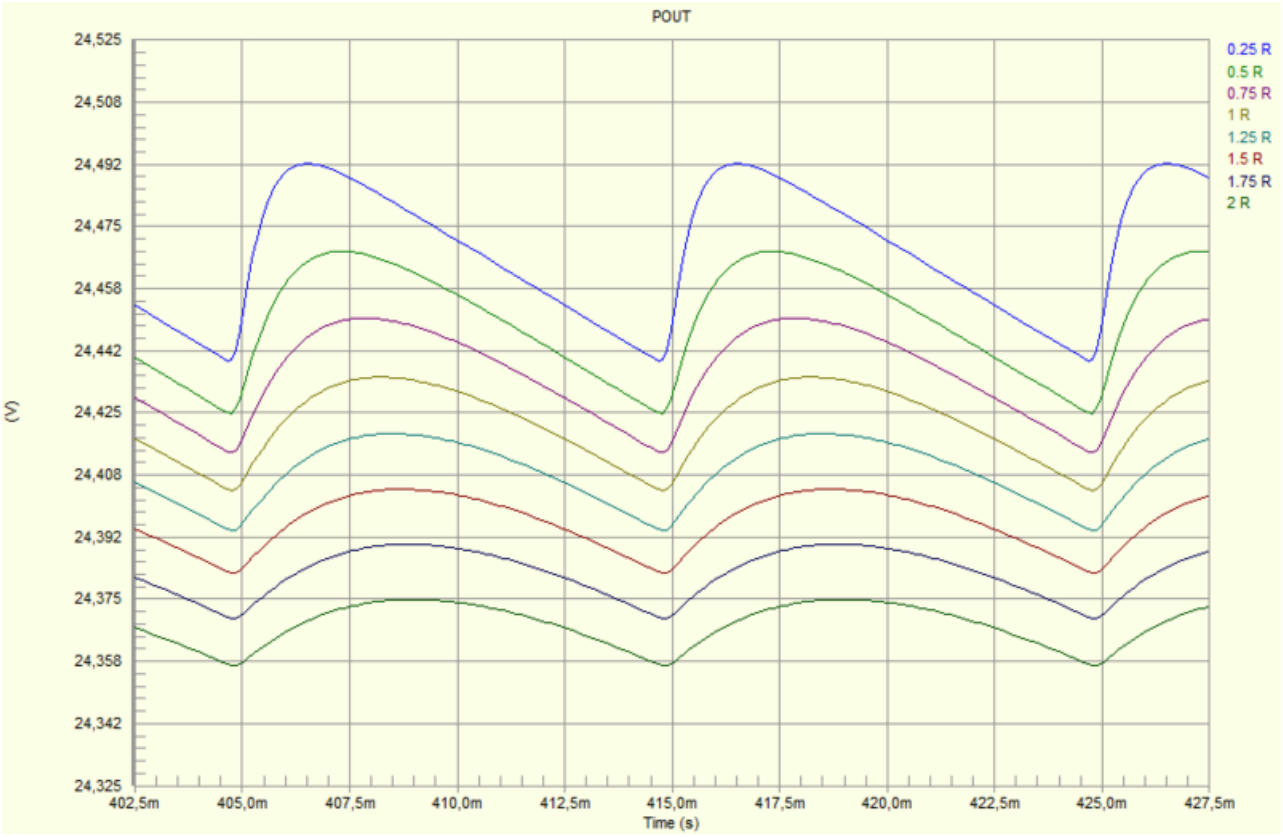
# ANNEXE 8

Influence de la valeur capacitive d'entrée du filtre d'alimentation



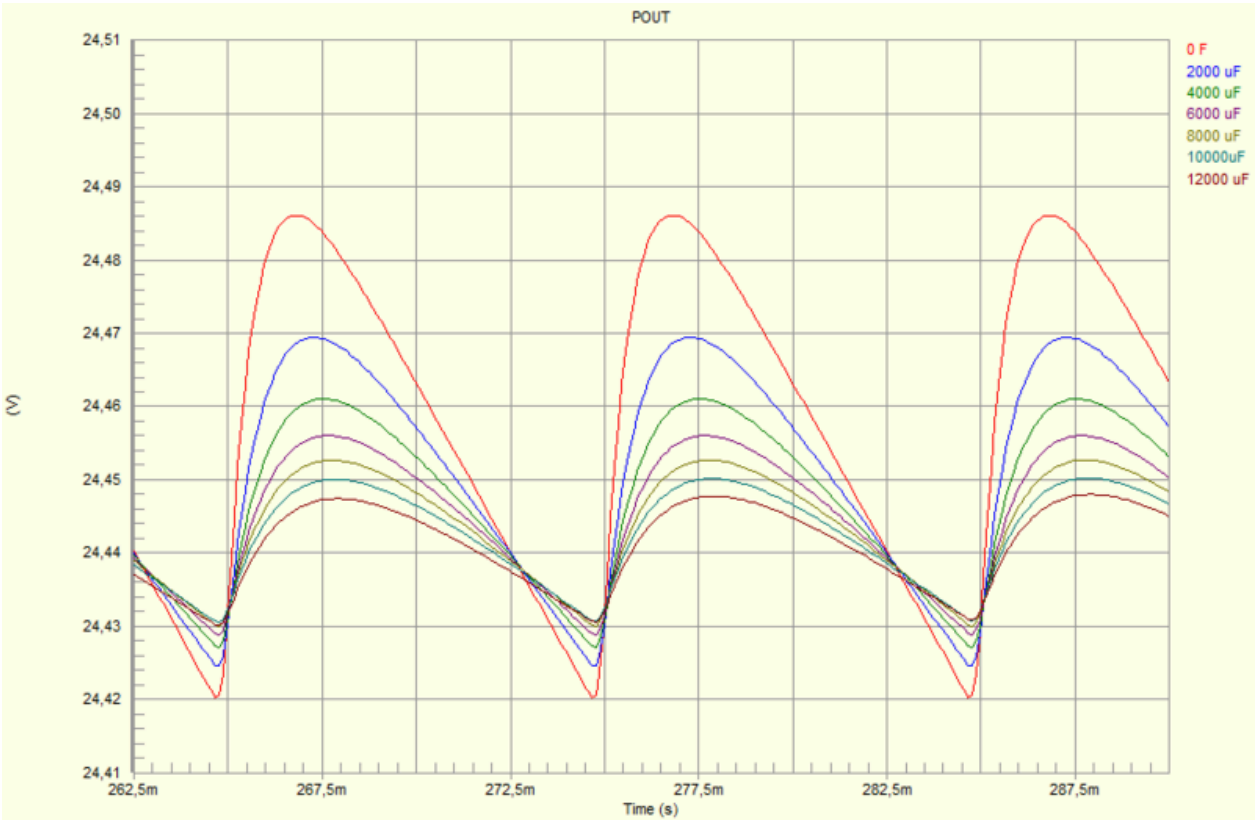
# ANNEXE 9

Influence de la valeur résistive du filtre d'alimentation



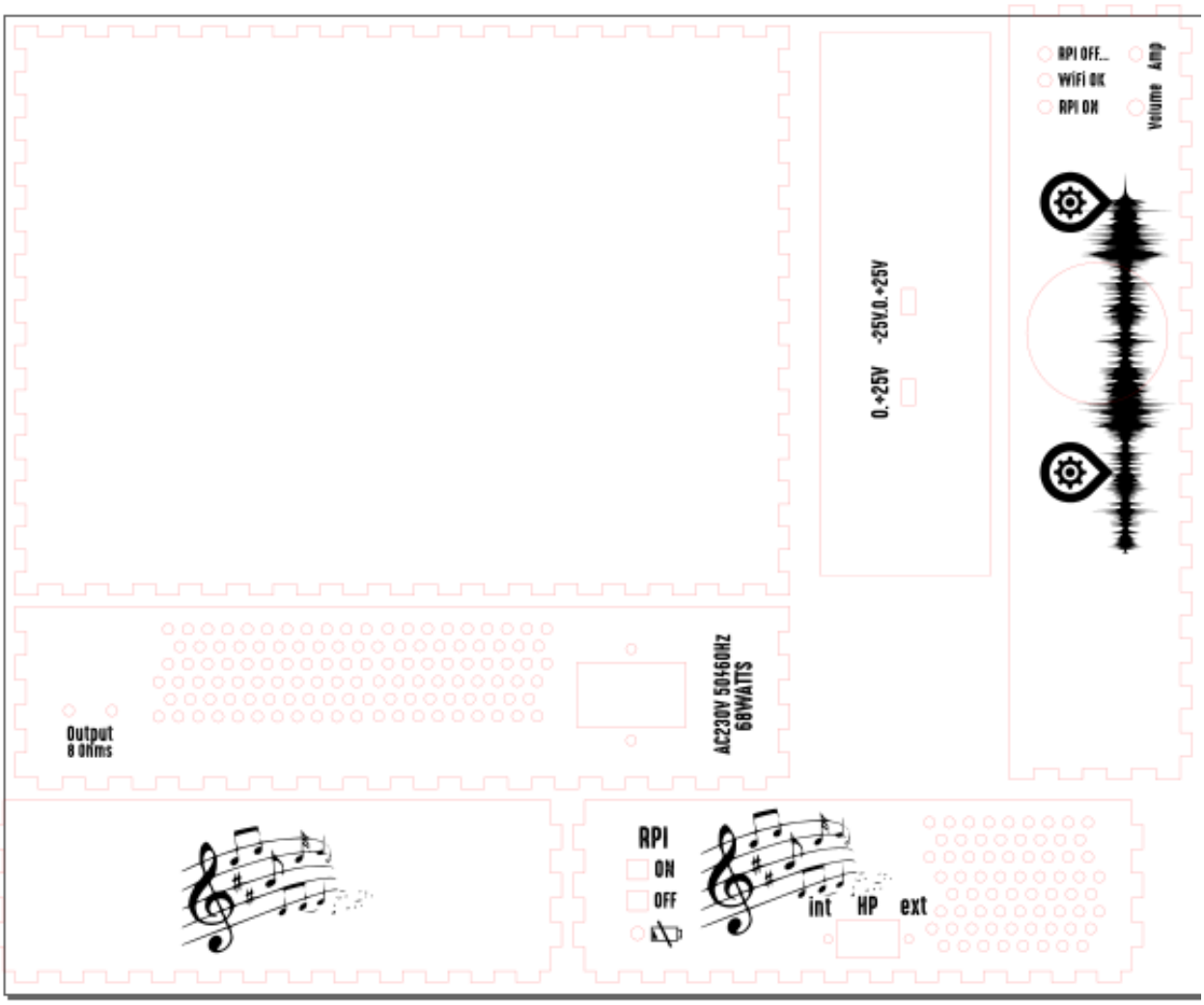
# ANNEXE 10

Influence de la valeur capacitive de sortie du filtre d'alimentation



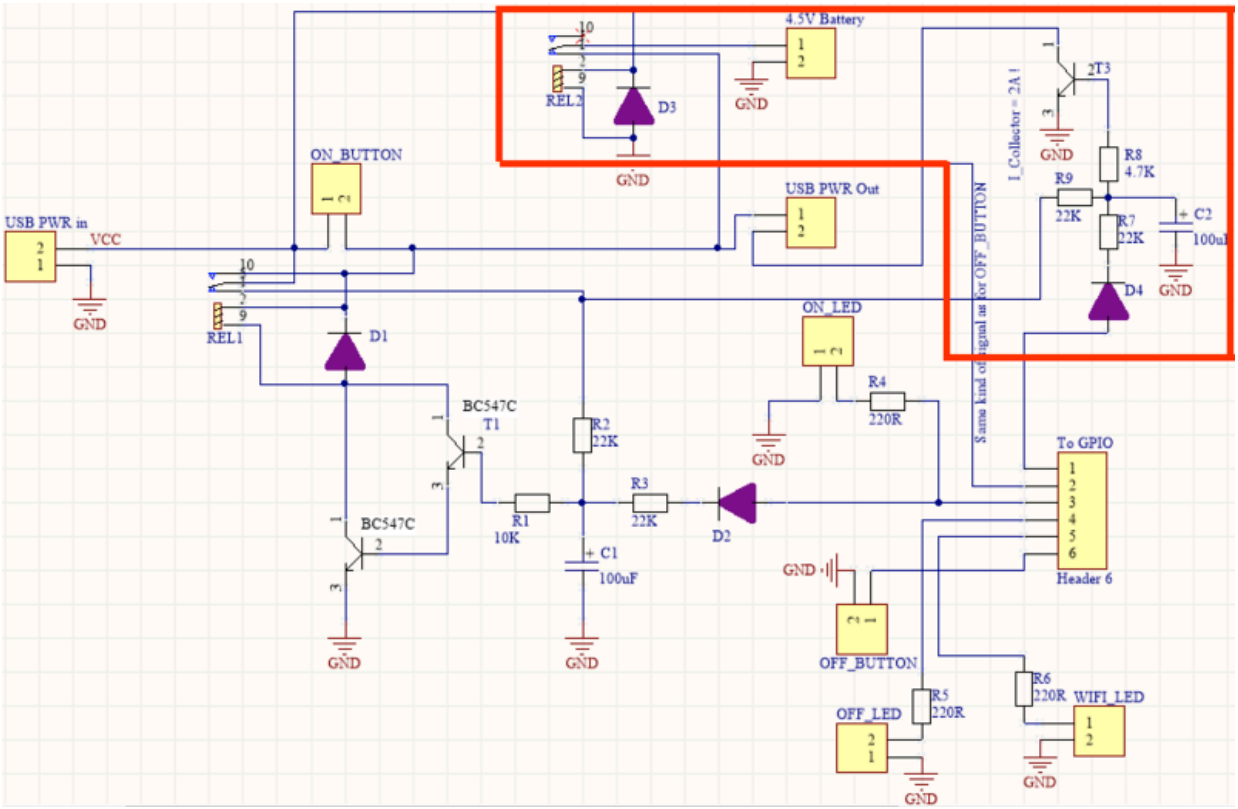
# ANNEXE 11

Fichier .svg pour la réalisation du boîtier



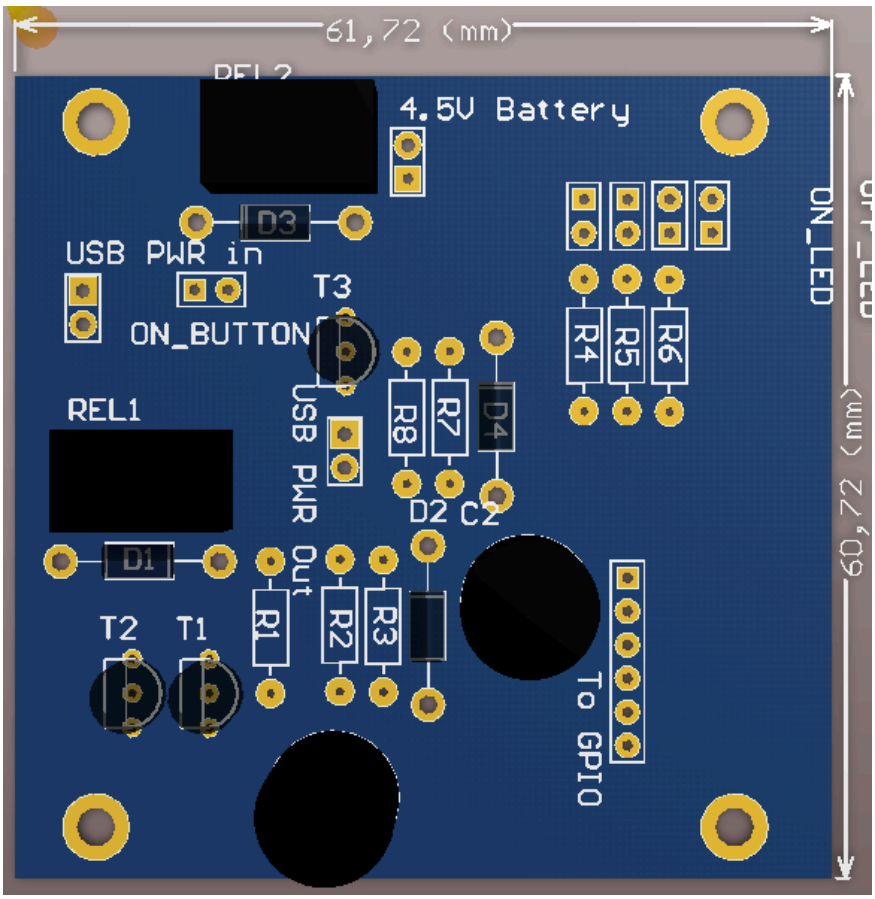
# ANNEXE 12

Circuit d'extinction propre de la Raspberry Pi



## ANNEXE 13

PCB du circuit d'extinction propre de la Raspberry Pi



## ANNEXE 14

Boîtier à grande modularité